MLPotential Manual

Amsterdam Modeling Suite 2026.1

www.scm.com

Apr 02, 2026

CONTENTS

CHAPTER

ONE

GENERAL

The MLPotential engine in the Amsterdam Modeling Suite can calculate the potential energy surface using several different types of machine learning (ML) potentials. To use ML potentials, first *install them* (page 5) separately.

The supported models can be found on the page *Models & Backends* (page 9).

1.1 Quickstart guide

To set up a simple MLPotential job using the **graphical user interface**, see the

- ANI-1ccx Thermochemistry tutorial
- M3GNet tutorial
- All MLPotential GUI tutorials

There are also *command-line examples* (page 25) and Python examples.

1.2 What's new in AMS2026.1?

- New *models* (page 9):
  - eSEN-S-Con-OMol is a pre-trained model based on DFT (ωB97M-V/def2-TZVPD) data from the OMol25 dataset, which contains data from diverse chemistry disciplines including biochemistry, electrochemistry, and organic and inorganic chemistry with all of the first 83 elements represented.
  - MACE-MP-0 is a pre-trained foundation model for materials chemistry, parameterized for 89 chemical elements. It is available in three sizes (small/medium/large) which balance accuracy vs. compute.
  - MACE-MPA-0 is trained on a larger dataset with additional crystal structures, and improves accuracy compared to MACE-MP-0.
  - UMA-S-1.2-OC20 is a pre-trained model based on DFT (RPBE) data, with training data comprising >100 million calculations of small molecules adsorbed on catalyst surfaces formed from materials in the Materials Project.
  - UMA-S-1.2-OC22 is a pre-trained model based on DFT (PBE/PBE+U) data, with training data comprising comprising ~10 million calculations of small molecules adsorbed on catalyst surfaces formed from unary and binary oxide materials in the Materials Project.
  - UMA-S-1.2-OC25 is a pre-trained model based on DFT (RPBE+D3) data, with training data comprising comprising ~8 million calculations of solvents, ions and electrolytes with catalyst surfaces formed from materials in the Materials Project.

  - UMA-S-1.2-ODAC is a pre-trained model based on DFT (PBE+D3) data, with training data comprising >10 million calculations of CO2/H2O molecules adsorbed in Metal Organic Frameworks sampled from various open databases like CoreMOF.
  - UMA-S-1.2-OMat is a pre-trained model based on DFT (PBE/PBE+U) data, with training data comprising >100 million calculations of inorganic materials collected from many open databases like Materials Project and Alexandria, and randomly sampled far from equilibria.
  - UMA-S-1.2-OMC is a pre-trained model based on DFT (PBE+D3) data, with training data comprising ~25 million calculations of organic molecular crystals from random packing of OE62 structures into various 3D unit cells.
  - UMA-S-1.2-OMol is a pre-trained model based on DFT (wB97M-V/def2-TZVPD) data, with training data comprising over 100 million calculations covering small molecules, biomolecules, metal complexes, electrolytes and polymers.
- Train custom MACE models with ParAMS and Simple Active Learning (and use them in the MLPotential engine)
- All models and backends installable using AMSpackages
- Removed: The SchNetPack (1.0.0) and sGDML (0.4.4) backends. To continue using these models, migrate to Engine ASE.

1.3 What's new in AMS2025.1?

- Improved support for external ML potentials (not included with AMS but available online) through Engine ASE.
- Deprecated: The SchNetPack (1.0.0) and sGDML (0.4.4) backends. To continue using these models, migrate to Engine ASE.

1.4 What's new in AMS2024.1?

- New *models* (page 9): AIMNet2-B973c and AIMNet2-wB97MD3. These are suitable for molecular systems containing H, B, C, N, O, F, Si, P, S, Cl, As, Se, Br, I. These are currently the only ML potential models that support charged systems (ions), and that predict atomic charges and dipole moments and that give IR intensities when calculating normal modes.
- Train custom M3GNet models with ParAMS and Simple Active Learning (and use them in the MLPotential engine)
- *Auto-detection* (page 23) of GPU.
- When using the ANI (or AIMNet2) models, the `mlpotential.txt` file is no longer produced, but the engine uncertainty (standard deviation of committee prediction) is written to the standard output and stored on the binary .rkf results files.

1.5 What's new in AMS2023.1?

- New model: M3GNet-UP-2022 based on M3GNet. This is a universal potential (UP) that can be used for the entire periodic table of elements up to, but excluding, Curium (Cm, 96).
- New backend: M3GNet
- PiNN is no longer a backend in MLPotential, but you can use it through Engine ASE.

1.6 Theory of ML potentials

With machine learning potentials, it is possible to quickly evaluate the energies and forces in a system with close to first-principles accuracy. Machine learning potentials are fitted (trained, parameterized) to reproduce reference data, typically calculated using an ab initio or DFT method. Machine learning potentials are sometimes referred to as machine learning force fields, or as interatomic potentials based on machine learning.

Several types of machine learning potentials exist, for example neural-network-based methods and kernel-based methods.

Several types of **neural network potentials** exist. It is common for such potentials to calculate the total energy as a sum of atomic contributions. In a **high-dimensional neural network potential** (HDNNP), as proposed by Behler and Parrinello[1], each atomic contribution is calculated by means of a feed-forward neural network, that takes in a representation of the chemical environment around the atom as input. This representation, or atomic environment **descriptor** or **fingerprint**, consists of a vector of rotationally, translationally, and permutationally invariant functions known as **atom-centered symmetry functions** (ACSF).

Graph convolutional neural network potentials (GCNNPs), or **message-passing network neural potentials**, similarly construct the total energy by summing up atomic contributions, but the appropriate representations of local atomic chemical environments are learned from the reference data.

Kernel-based methods make predictions based on how similar a system is to the systems in the training set.

There are also other types of machine learning potentials. For more detailed information, see for example references[2] and[3].

1.7 Support

SCM provides technical (non-scientific) support for installation and running simulations via the AMS driver.

See also: *Frequently Asked Questions* (page 33)

1.8 Technical information

Each of the supported backends can be used as ASE (Atomic Simulation Environment) calculators (https://wiki.fysik.dtu.dk/ase/index.html). The MLPotential engine is an interface to those ASE calculators. The communication between the AMS driver and the backends is implemented with a named pipe interface. The MLPotential engine launches a Python script, `ase_calculators.py`, which initializes the ASE calculator. The exact command that is executed is written as `WorkerCommand` in the output.

[1] J. Behler, M. Parrinello. Phys. Rev. Lett. 98 (2007) 146401 https://doi.org/10.1103/PhysRevLett.98.146401
[2] J. Behler. J. Chem. Phys. 145 (2016) 170901. https://doi.org/10.1063/1.4966192
[3] T. Mueller, A. Hernandez, C. Wang. J. Chem. Phys. 152 (2020) 050902. https://doi.org/10.1063/1.4966192

1.9 References

CHAPTER

TWO

INSTALLATION & UNINSTALLATION

Tip: If AMS does not support your preferred ML potential, you may be able to create a Python environment yourself and use it through engine ASE.

The Amsterdam Modeling Suite requires the installation of additional Python environments to run the machine learning potential backends.

If you set up an MLPotential job via the **graphical user interface**, you will be asked to install the environments if they have not been installed already when you save your input. You can also use the package manager. A **command-line installation tool** can also be used, for instance to install the torchani backend:

```
"$AMSBIN"/amspackages install torchani-cpu
```

You can use the command line installer to install these environments on a remote system, so that you can seamlessly run MLPotential jobs also on remote machines.

The packages are installed into an isolated Python environment and do not affect any other Python installation on the system. For the installation, an Internet connection is required, unless you have configured the AMS package manager for offline use.

To **uninstall** a package, e.g. `torchani`, run:

```
"$AMSBIN"/amspackages remove torchani-cpu
```

2.1 Installing GPU-enabled backends using AMSpackages

Note: These instructions are suitable for AMS2026.101 and later. See here (https://www.scm.com/doc.2025/MLPotential/Installation.html#installing-gpu-enabled-backends-using-amspackages) for older versions of AMS.

Various versions of the ML potential environments may be available through the AMSpackages, with different system dependencies such as GPU drivers. The type of the package is indicated by the name and package-id postfix.

For example, for MACE you will see the following entries:

```
$ "$AMSBIN"/amspackages list --format long | grep -A 3 "MACE Environment"
Package : MACE Environment (v1, CPU Only)
[id]: mace-cpu
Description: This is the environment for the MACE machine learning potential
```

(continues on next page)

(continued from previous page)

```
(https://github.com/ACEsuit/mace), for CPU only on all platforms.
--
Package : MACE Environment (v1, CUDA 12.8)
[id]: mace-cu128
Description: This is the environment for the MACE machine learning potential
(https://github.com/ACEsuit/mace), for CUDA 12.8.
```

Note that CUDA-enabled versions are currently only available for the Linux platform.

Then select the version to install. Selecting `-cpu` variants will use CPU-only versions of PyTorch and TensorFlow, while `-cu128` will use the CUDA-enabled 12.8 versions. CUDA-enabled environments are typically much larger in size as the required drivers are included, but allow usage of the GPU which can lead to significant speedups when using ML models. Note that the CUDA-enabled environments can also be used to run on CPU only, and so only one version (either CPU only or CUDA) should be installed.

If a suitable version of the environment is not available for your needs, you may be able to create a Python environment yourself and use it through engine ASE.

2.2 Debugging installation and available resources

A tool is provided to investigate the current installation of ML backends and frameworks. This tool will also report the resources that would be found by AMS if a calculation was performed with default settings.

The tool is used as follows:

```
$AMSBIN/amspython $AMSHOME/Utils/check_ml_backends.py
```

Example output:

```
ML Environment Summary:

AIMNet2:
Available Models: AIMNet2-B973c, AIMNet2-wB97MD3
Installed: not installed
Environments: aimnet2-cpu, aimnet2-cu128

FAIRChem:
Available Models: Custom, eSEN-S-Con-OMol, eSEN-Sm-Conserving, UMA-S-1.2-OC20, UMA-S-
↪1.2-OC22, UMA-S-1.2-OC25, UMA-S-1.2-ODAC, UMA-S-1.2-OMat, UMA-S-1.2-OMC, UMA-S-1.2-
↪OMol, UMA-S-1P2
Installed: not installed
Environments: fairchem-cpu, fairchem-cu128

M3GNet:
Available Models: Custom, M3GNet-UP-2022, MP-2021.2.8-EFS
Installed: installed (m3gnet)
Environments: m3gnet

MACE:
Available Models: Custom, MACE-MP-0-Large, MACE-MP-0-Medium, MACE-MP-0-Small, MACE-
↪MPA-0
Installed: installed (mace-cpu)
Environments: mace-cpu, mace-cu128

NequIP:
```

(continues on next page)

(continued from previous page)

```
Available Models: Custom
Installed: not installed
Environments: nequip-cpu, nequip-cu128

TorchANI:
Available Models: ANI-1ccx, ANI-1x, ANI-2x, Custom
Installed: not installed
Environments: torchani-cpu, torchani-cu128

Testing Installed Backends:
M3GNet:
Dependencies installed: True

###################TensorFlow setup##################
TensorFlow 2.9.1-cpu found the following devices:
PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU')
NumThreads was not specified in the MLPotential engine block, so TensorFlow will use␣
↪all available cores.
#####################################################

MACE:
Dependencies installed: True

####################PyTorch setup###################
PyTorch 2.8.0 found the following devices:
Number of threads were not limited, using all available CPU cores.
Using CPU only.
#####################################################
```

If there are any issues, before contacting support (https://support.scm.com), please run the following command:

```
$AMSBIN/amspython $AMSHOME/Utils/check_ml_backends.py --debug
```

and then report the output with your question.

This will give us additional details on why a framework or backend was not considered installed and report potential issues in the environment.

CHAPTER

THREE

MODELS & BACKENDS

3.1 Included (pre-parameterized) models

A **model** is the combination of a functional form with a set of parameters. A number of pre-parameterized models can be selected in AMS.

Model	Primary Domain	Backend
AIMNet2-B973c (page 10)	(Charged) Organic Molecules, Organohalides	AIMNet2
AIMNet2-wB97MD3 (page 10)	(Charged) Organic Molecules, Organohalides	AIMNet2
ANI-1ccx (page 11)	Organic Molecules	TorchANI
ANI-1x (page 11)	Organic Molecules	TorchANI
ANI-2x (page 11)	Organic Molecules	TorchANI
eSEN-S-Con-OMol (page 12)	Organic Molecules, Biomolecules, Metal Complexes, Electrolytes	FAIRChem
M3GNet-UP-2022 (page 14)	Materials	M3GNet
MACE-MP-0-Large (page 15)	Materials	MACE
MACE-MP-0-Medium (page 15)	Materials	MACE
MACE-MP-0-Small (page 15)	Materials	MACE
MACE-MPA-0 (page 15)	Materials	MACE
UMA-S-1.2-OC20 (page 16)	Adsorbates on Catalytic Surfaces	FAIRChem
UMA-S-1.2-OC22 (page 16)	Adsorbates on Catalytic Oxide Surfaces	FAIRChem
UMA-S-1.2-OC25 (page 16)	Solvents	Ions and Electrolytes with Catalytic Surfaces FAIRChem
UMA-S-1.2-ODAC (page 16)	CO_2/H_2O adsorbed in MOFs	FAIRChem
UMA-S-1.2-OMat (page 16)	Inorganic Materials	FAIRChem
UMA-S-1.2-OMC (page 16)	Organic Molecular Crystals	FAIRChem
UMA-S-1.2-OMol (page 16)	Organic Molecules, Biomolecules, Metal Complexes, Electrolytes	FAIRChem

3.1.1 AIMNet2 Models

AIMNet2 (Atoms In Molecules Network) is a neural network potential designed for accurate predictions of molecular geometries and reaction energies for both neutral and gas-phase charged organic molecules. It incorporates explicit long-range electrostatics and dispersion contributions and uses charge equilibration within the message passing framework, enabling improved performance for systems where long-range interactions are important.

Two pre-trained models are available in AMS: AIMNet2-wB97MD3 and AIMNet2-B973c[1]. The difference between them is that AIMNet2-wB97MD3 is trained to more expensive and accurate ωB97M-D3/def2-TZVPP DFT reference data, whereas AIMNet2-B973c uses only B97-3c reference data.

Best for

- Fast calculations of small, drug-like molecules to routine DFT level accuracy
- Predictions of atomic charges and dipole moments

Limitations

- Does not support periodic systems
- Restricted to 14 elements (H, B, C, N, O, F, Si, P, S, Cl, As, Se, Br, I)

Training data

Dataset	20 million conformers, including charged species, distilled from an initial pool of 120 million
Reference methods	B97-3c (AIMNet2-B973c), ωB97M-D3/def2-TZVPP (AIMNet2-wB97MD3)
Included elements	H, B, C, N, O, F, Si, P, S, Cl, As, Se, Br, I

Notes

- Predictions from AIMNet2-B973c are calculated from **committees** (ensembles), meaning that the final prediction is an average over multiple independently trained neural networks.

Examples

An example AMS input file for the geometry optimization of chloromethane with AIMNet2-wB97MD3 is as follows:

```
#!/bin/sh

export NSCM=1

"$AMSBIN/ams" --delete-old-results << eor
Task GeometryOptimization

System
    Atoms
        C -0.13367473200681762 0.0032536323351088866 0.001112839342431487
```

(continues on next page)

[1] D. M. Anstine et al., Chem Sci. 16 (2025) 10228–10244. https://doi.org/10.1039/D4SC08572H

(continued from previous page)

```
        Cl 1.6437227908091656 -0.04000788304685571 -0.013683802158888616
        H -0.5076353557849584 -0.4795638958097295 0.9272936171699165
        H -0.5242024748732897 -0.5409765935860863 -0.8832353361874874
        H -0.47821022814410086 1.0572947401075619 -0.03148731816597247
    End
End

Engine MLPotential
    Model AIMNet2-wB97MD3
EndEngine
eor
```

See also *further examples* (page 25).

3.1.2 ANI Models

ANI (Accurate NeurAl networK engINe for Molecular Energies) is a family of neural network potentials for efficient and accurate prediction of molecular geometries, vibrational frequencies, and reaction energies for gas-phase organic molecules.

Three pre-trained models are available in AMS: ANI-1x[2], ANI-1ccx[3], and ANI-2x[4]. ANI-1x and ANI-2x are trained to ωB97X/6-31G(d) DFT reference data, while ANI-1ccx is trained using transfer learning to target coupled-cluster quality reference energies CCSD(T)*/CBS.

Best for

- Very fast calculations of organic molecules (ANI-1x, ANI-1ccx), also including light halogens and sulfur (ANI-2x)

Limitations

- Restricted to a small set of light elements
- Does not support charged systems

Training data

Dataset	5.5 million conformers of small organic molecules (ANI-1x), supplemented with additional conformers containing light elements to make 8.9 million in total (ANI-2x)
Reference methods	ωB97X/6-31G(d) (ANI-1x, ANI-2x), DLPNO-CCSD(T)/CBS (ANI-1ccx)
Included elements	H, C, N, O (ANI-1x, ANI-1ccx), F, S, Cl (ANI-2x)

[2] J. S. Smith et al., J. Chem. Phys. 148 (2018) 241733. https://doi.org/10.1063/1.5023802
[3] J. S. Smith et al., Nat. Commun. 10 (2019) 2903. https://doi.org/10.1038/s41467-019-10827-4
[4] C. Devereux et al., J. Chem. Theory Comput. 16 (2020) 4192-4202. https://doi.org/10.1021/acs.jctc.0c00121

Notes

- Predictions from ANI models are calculated from **committees** (ensembles), meaning that the final prediction is an average over multiple independently trained neural networks.

Examples

An example AMS input file for the geometry optimization of methane with ANI-1ccx is as follows:

```
#!/bin/sh

export NSCM=1

"$AMSBIN/ams" --delete-old-results << eor
Task GeometryOptimization

System
    Atoms
        C 2.9166023165223268e-09 6.006194625417838e-09 7.204071579427031e-10
        H 0.5389120953335456 0.7623581289443943 -0.5992945752094279
        H 0.7312440919932264 -0.5966159238699776 0.5831823400796503
        H -0.5671285553273658 -0.6703024318238477 -0.6781076404295054
        H -0.7030276349160114 0.5045602207432393 0.6942198748388725
    End
End

Engine MLPotential
    Model ANI-1ccx
EndEngine
eor
```

3.1.3 eSEN Models

eSEN (Equivariant Smooth Energy Network) is a neural network potential developed for highly accurate computation of energies and forces for molecules with diverse chemistries. This includes the modeling of charged and open-shell systems.

In AMS, the pre-trained eSEN-S-Con-OMol[5] (con=conserving) model is provided, which is trained on the OMol25 dataset, designed to cover a very broad range of organic and bio-relevant molecules, including metal complexes and electrolytes.

Best for

- Highly accurate calculations of diverse organic and bio-relevant molecules
- Molecular systems containing main-group elements and many heavier elements up to Bi
- Inclusion of system charge and spin-multiplicity

[5] D. S. Levine et al., arXiv:2505.08762 (2025). https://arxiv.org/abs/2505.08762

Limitations

- Not intended for calculations on periodic inorganic bulk materials

Training data

Dataset	OMol25 dataset comprising over 100 million structures covering small molecules, biomolecules, metal complexes, and electrolytes
Reference methods	wB97M-V/def2-TZVPD, including non-local dispersion
Included elements	Elements from H .. Bi

Notes

- All training data is aperiodic, so any periodic systems should be treated with some caution
- eSEN models are made accessible for commercial and non-commercial use under the permissive FAIRChem license (https://huggingface.co/facebook/OMol25/blob/main/LICENSE), which applies when using these models

Examples

An example AMS input file for the geometry optimization of methane with eSEN-S-Con-OMol is as follows:

```
#!/bin/sh

export NSCM=1

"$AMSBIN/ams" --delete-old-results << eor
Task GeometryOptimization

System
    Atoms
        C 2.9166023165223268e-09 6.006194625417838e-09 7.204071579427031e-10
        H 0.5389120953335456 0.7623581289443943 -0.5992945752094279
        H 0.7312440919932264 -0.5966159238699776 0.5831823400796503
        H -0.5671285553273658 -0.6703024318238477 -0.6781076404295054
        H -0.7030276349160114 0.5045602207432393 0.6942198748388725
    End
End

Engine MLPotential
    Model eSEN-S-Con-OMol
EndEngine
eor
```

3.1.4 M3GNet Models

M3GNet (Materials based on Graph Neural Networks with three-body interactions) is an interatomic potential designed for atomistic simulations of periodic materials.

The M3GNet-UP-2022[6] model available in AMS is intended to be "universal", i.e., applicable to a broad range of materials containing elements from across the periodic table, although the training data is primarily made up of crystal data from inorganic materials from the Materials Project[7].

Best for

- Fast calculations of inorganic crystalline materials
- Periodic systems (bulk solids, surfaces, interfaces)

Limitations

- Not designed for accurately modeling small organic molecules or biomolecules

Training data

Dataset	187k structures from 63k materials from the Materials Project
Reference methods	PBE, PBE+U
Included elements	Elements from H .. Pu (except Po .. Ra)

Notes

- M3GNet-UP-2022 can be fine-tuned with ParAMS

Examples

An example AMS input file for the geometry optimization of methane with M3GNet-UP-2022 is as follows:

```
#!/bin/sh

export NSCM=1

"$AMSBIN/ams" --delete-old-results << eor
Task GeometryOptimization

System
    Atoms
        Na 0.0 0.0 0.0
        Cl 2.815 2.815 2.815
    End
    Lattice
        0.0 2.815 2.815
        2.815 0.0 2.815
```

(continues on next page)

[6] C. Chen, S. P. Ong. Nature Computational Science 2, 718–728 (2022). https://doi.org/10.48550/arXiv.2202.02450
[7] A. Jain et al., APL Materials 1 (2013) 011002. https://doi.org/10.1063/1.4812323

(continued from previous page)

```
        2.815 2.815 0.0
    End
End

Engine MLPotential
    Model M3GNet-UP-2022
EndEngine
eor
```

See also *further examples* (page 26).

3.1.5 MACE Models

MACE (Message Passing Atomic Cluster Expansion)[8] is a family of equivariant neural network interatomic potentials designed for accurate prediction of energies and forces in atomistic simulations.

MACE-MP-0 and MACE-MPA-0 models[9] available in AMS are pre-trained foundation potentials targeting inorganic materials chemistry and are intended for periodic systems. For MACE-MP-0, multiple model sizes are provided (Small, Medium, Large), offering a trade-off between computational cost and accuracy. MACE-MPA-0 is equivalent in size to "Medium", but is trained on a larger dataset with additional crystal structures for improved accuracy.

Best for

- Accurate periodic calculations of inorganic materials
- Tuning speed/accuracy via Small/Medium/Large variants

Limitations

- Not designed for accurately modeling small organic molecules or biomolecules

Training data

Dataset	Materials Project MPtraj dataset comprising 1.58 million structures from 146k materials (MACE-MP-0), supplemented with the sAlex dataset comprising a further 10.4 million structures from 3.23 million materials (MACE-MPA)
Reference methods	PBE+U
Included elements	Elements from H .. Pu (except Po .. Ra)

[8] I. Batatia et al., Advances in Neural Information Processing Systems (2022). https://openreview.net/forum?id=YPpSngE-ZU
[9] I. Batatia et al., arXiv:2401.00096 (2023). https://arxiv.org/abs/2401.00096

Notes

- MACE-MP-0 and MACE-MPA-0 can be fine-tuned with ParAMS

Examples

An example AMS input file for the geometry optimization of methane with MACE-MPA-0 is as follows:

```
#!/bin/sh

export NSCM=1

"$AMSBIN/ams" --delete-old-results << eor
Task GeometryOptimization

System
    Atoms
        Na 0.0 0.0 0.0
        Cl 2.815 2.815 2.815
    End
    Lattice
        0.0 2.815 2.815
        2.815 0.0 2.815
        2.815 2.815 0.0
    End
End

Engine MLPotential
    Model MACE-MPA-0
EndEngine
eor
```

See also *further examples* (page 26).

3.1.6 UMA Models

UMA (Universal Model for Atoms)[10] is a foundation neural network potential with models trained on large-scale atomistic datasets spanning molecules, materials, surfaces, adsorption systems, and molecular crystals, which are intended to provide broad transferability across diverse chemistry.

Several pre-trained UMA variants are available in AMS. Each variant is specialized towards a particular training domain while retaining a shared underlying model architecture.

These include:

- UMA-S-1.2-OC20: adsorbates on catalytic surfaces (solid-gas interfaces)
- UMA-S-1.2-OC22: adsorbates on catalytic oxide surfaces (solid-gas interfaces)
- UMA-S-1.2-OC25: solvent layers, ions and electrolytes with catalytic surfaces (solid-liquid interfaces)
- UMA-S-1.2-ODAC: CO_2/H_2O adsorbed in MOFs
- UMA-S-1.2-OMat: inorganic materials
- UMA-S-1.2-OMC: organic molecular crystals

[10] B. M. Wood et al., arXiv:2506.23971 (2025). https://arxiv.org/abs/2506.23971

- UMA-S-1.2-OMol: organic molecules, biomolecules, metal complexes, electrolytes

Best for

- High accuracy calculations on a broad range of systems including molecules (OMol) and inorganic materials (OMat) with diverse chemistry
- Charged / open-shell molecules including radicals (OMol)
- Adsorption and surface chemistry with solid-gas interfaces (OC20), oxide materials (OC22) and electrocatalysts with solid-liquid interfaces (OC25)
- Porous framework adsorption (ODAC)
- Organic molecular crystals (OMC)

Limitations

- Predictions are best within the dominant chemistry represented in the chosen UMA variant
- Relatively computationally expensive compared to other, more targeted models

Training data

Model	UMA-S-1.2-OC20
Dataset	OC20 dataset comprising >100 million calculations of small molecules adsorbed on catalyst surfaces formed from materials in the Materials Project
Reference methods	RPBE, no dispersion
Included elements	Elements H .. Cl .. Bi (except other grp. 7, grp. 8 or Mg, Ba, Ln)

Model	UMA-S-1.2-OC22
Dataset	OC22 dataset comprising ~10 million calculations of small molecules adsorbed on catalyst surfaces formed from unary and binary oxide materials in the Materials Project
Reference methods	PBE/PBE+U, no dispersion
Included elements	Elements Li, Be, O, Na, Mg, Al, Si, K .. Ce, Lu .. Bi (except grp. 7, grp. 8, As, Tc, Te)

Model	UMA-S-1.2-OC25
Dataset	OC25 dataset comprising ~8 million calculations of solvents, ions and electrolytes with catalyst surfaces formed from materials in the Materials Project
Reference methods	RPBE+D3
Included elements	Elements H .. Pm, Hf .. Bi

Model	UMA-S-1.2-ODAC
Dataset	ODAC23 dataset comprising >10 million calculations of CO_2/H_2O molecules adsorbed in Metal Organic Frameworks sampled from various open databases like CoreMOF
Reference methods	PBE+D3
Included elements	Elements H, Li .. Np (except grp. 8, K, Rb, Tc, In, Pm, Yb, Ta, Os, Ir, Tl, Pb, Po .. Ac, Pa)

Model	UMA-S-1.2-OMat
Dataset	OMat24 dataset comprising >100 million calculations of inorganic materials collected from many open databases like Materials Project and Alexandria, and randomly sampled far from equilibria
Reference methods	PBE/PBE+U, no dispersion
Included elements	Elements H .. Pu (except Po .. Ra)

Model	UMA-S-1.2-OMC
Dataset	OMC25 dataset comprising ~25 million calculations of organic molecular crystals from random packing of OE62 structures into various 3D unit cells
Reference methods	PBE+D3
Included elements	H, B, C, N, O, F, Si, P, S, Cl, Br, I

Model	UMA-S-1.2-OMol
Dataset	OMol25 dataset comprising over 100 million structures covering small molecules, biomolecules, metal complexes, and electrolytes. Supplemented with OPoly26 dataset comprising ~6.35 million structures of various polymeric systems including traditional polymers, fluoropolymers, optical polymers, peptoids, lipids and polymer electrolytes.
Reference methods	wB97M-V/def2-TZVPD, including non-local dispersion
Included elements	Elements from H .. Bi

Notes

- A UMA variant should be selected based on the dominant system type
- Only UMA-S-1.2-OMol supports charged systems/spin multiplicity
- All UMA-S-1.2-OMol training data is aperiodic, so any periodic systems should be treated with some caution
- UMA models are made accessible for commercial and non-commercial use under the permissive FAIRChem license (https://huggingface.co/facebook/UMA/blob/main/LICENSE), which applies when using these models

Examples

An example AMS input file for the geometry optimization of methane with UMA-S-1.2-OMol is as follows:

```
#!/bin/sh

export NSCM=1

"$AMSBIN/ams" --delete-old-results << eor
Task GeometryOptimization

System
    Atoms
        C 2.9166023165223268e-09 6.006194625417838e-09 7.204071579427031e-10
        H 0.5389120953335456 0.7623581289443943 -0.5992945752094279
        H 0.7312440919932264 -0.5966159238699776 0.5831823400796503
        H -0.5671285553273658 -0.6703024318238477 -0.6781076404295054
        H -0.7030276349160114 0.5045602207432393 0.6942198748388725
    End
End

Engine MLPotential
    Model UMA-S-1.2-OMol
EndEngine
eor
```

An example AMS input file for the geometry optimization of sodium chloride with UMA-S-1.2-OMat is as follows:

```
#!/bin/sh

export NSCM=1

"$AMSBIN/ams" --delete-old-results << eor
Task GeometryOptimization

System
    Atoms
        Na 0.0 0.0 0.0
        Cl 2.815 2.815 2.815
    End
    Lattice
        0.0 2.815 2.815
        2.815 0.0 2.815
        2.815 2.815 0.0
    End
End

Engine MLPotential
    Model UMA-S-1.2-OMat
EndEngine
eor
```

See also *further examples* (page 29).

`Model`

Type
Multiple Choice

Default value
ANI-2x

Options
[Custom, AIMNet2-B973c, AIMNet2-wB97MD3, ANI-1ccx, ANI-1x, ANI-2x, eSEN-S-Con-OMol, M3GNet-UP-2022, MACE-MP-0-Large, MACE-MP-0-Medium, MACE-MP-0-Small, MACE-MPA-0, UMA-S-1.2-OC20, UMA-S-1.2-OC22, UMA-S-1.2-OC25, UMA-S-1.2-ODAC, UMA-S-1.2-OMat, UMA-S-1.2-OMC, UMA-S-1.2-OMol]

Description
Select a pre-parameterized or custom model.

AIMNet2-(wB97MD3/B973c): best for fast calculations of small, drug-like molecules; limited to aperiodic systems of 14 elements (H, B, C, N, O, F, Si, P, S, Cl, As, Se, Br, I).

ANI-(1x/1ccx/2x): best for very fast calculations of organic molecules; limited to elements H, C, N, O (ANI-1x/1ccx), F, S, Cl (ANI-2x).

eSEN-S-Con-OMol: best for highly accurate calculations of diverse organic and bio-relevant molecules; not intended for calculations on periodic inorganic bulk materials.

M3GNet-UP-2022: best for fast calculations of inorganic crystalline materials; not designed for accurately modeling small organic molecules or biomolecules.

MACE-MP-0-(Small/Medium/Large): best for accurate periodic calculations of inorganic materials; size trades speed/accuracy; not designed for accurately modeling small organic molecules or biomolecules. MACE-MPA-0 has improved accuracy vs MP-0.

UMA-S-1.2 variants: best for high accuracy calculations on a broad range of systems; choose from OC20 (adsorption and surface chemistry), OC22 (oxide catalysis), OC25 (electrocatalysis), ODAC (adsorption in porous frameworks), OMat (inorganic materials), OMC (organic molecular crystals), OMol (molecules, biomolecules, metal complexes, electrolytes); can be computationally expensive compared to other, more targeted models.

Set Custom to choose a backend and provide your own parameters.

3.2 Custom models (custom parameters)

Note: You can use Engine ASE to use any ASE calculator as the engine.

Note: You can use ParAMS to train your own ML potential parameters.

Set `Model` to **Custom** and specify which backend to use with the `Backend` option. In a typical case, you would have used that backend to train your own machine learning potential.

The backend reads the parameters, and any other necessary information (for example neural network architecture), from either a file or a directory. Specify the `ParameterFile` or `ParameterDir` option accordingly, with a path to the file or directory. Read the backend's documentation to find out which option is appropriate.

Example:

```
Engine MLPotential
    Backend MACE
    Model Custom
```

(continues on next page)

(continued from previous page)

```
    ParameterFile mace-custom.model
EndEngine
```

Backend

Type
Multiple Choice

Options
[FAIRChem, M3GNet, MACE, NequIP, TorchANI]

Description
The machine learning potential backend.

ParameterDir

Type
String

Default value
GUI name
Parameter directory

Description
Path to a set of parameters for the backend, if it expects to read from a directory.

ParameterFile

Type
String

Default value
Description
Path to a set of parameters for the backend, if it expects to read from a file.

3.3 Backends

Table 3.1: Backends supported by the MLPotential engine.

	FAIRChem	M3GNet	MACE	NequIP	TorchANI
Reference	Page 16, 10	Page 14, 6	Page 15, 8	11	12
Parameters from	ParameterFile	ParameterDir	ParameterFile	ParameterFile	ParameterFile
Included models	eSEN-S-Con-OMol, UMA-S-1.2-OC20, UMA-S-1.2-OC22, UMA-S-1.2-OC25, UMA-S-1.2-ODAC, UMA-S-1.2-OMat, UMA-S-1.2-OMC, UMA-S-1.2-OMol	M3GNet-UP-2022	MACE-MP-0, MACE-MPA-0	none	ANI-1x, ANI-2x, ANI-1ccx
ML framework	PyTorch 2.8.0	TensorFlow 2.9.1	PyTorch 2.8.0	PyTorch 2.8.0	PyTorch 2.8.0

Note: Technically, there is also an AIMNet2 backend but it can only be activated through the pre-parametrized models AIMNet2-B973c and AIMNet2-wB97MD3.

Note: Starting with AMS2023, PiNN[14] is only supported as a custom Calculator through Engine ASE[13].

Starting with AMS2026, SchNetPack[15] and sGDML[16] are also only supported as a custom Calculator through Engine ASE[Page 22, 13].

Note: If you use a custom parameter file with **TorchANI**, the model specified via `ParameterFile filename.pt` is loaded with `torch.load('filename.pt')['model']`, such that a forward call should be accessible via `torch.load('filename.pt')['model']((species, coordinates))`. The energy shifter is not read from custom parameter files, so the absolute predicted energies will be shifted with respect to the reference data, but this does not affect relative energies (e.g., reaction energies).

3.4 References

[11] S. Batzner et al., Nat. Commun. 13 (2022) 2453. https://doi.org/10.1038/s41467-022-29939-5
[12] X. Gao et al. J. Chem. Inf. Model (2020). https://doi.org/10.1021/acs.jcim.0c00451
[14] Y. Shao et al., J. Chem. Inf. Model. 60 (2020) 1184-1193. https://doi.org/10.1021/acs.jcim.9b00994
[13] https://wiki.fysik.dtu.dk/ase/index.html
[15] K. T. Schütt et al., J. Chem. Theory Comput. 15 (2019) 448-455. https://doi.org/10.1021/acs.jctc.8b00908
[16] S. Chmiela et al. Comp. Phys. Commun. 240 (2019) 38-45. https://doi.org/10.1016/j.cpc.2019.02.007

CHAPTER

FOUR

CPU & GPU (CUDA), PARALLELIZATION

If the GPU-enabled versions of the backends have been *installed* (page 5), then by default the calculation will try to auto-detect if there is a GPU available to run on, and if so use that. But you can also force it to run on a specific `Device`.

To limit the number of CPU threads, the `NumThreads` keyword can be used if the backend uses PyTorch as its machine learning framework. Alternatively, you can set the environment variable OMP_NUM_THREADS.

To use a CUDA-enabled GPU, ensure that a CUDA-enabled version of TensorFlow or PyTorch has been installed (see *Installation & Uninstallation* (page 5)).

If the software has problems detecting the GPU, see *Debugging installation and available resources* (page 6).

`Device`

Type
Multiple Choice

Default value

Options
[, cpu, cuda:0, cuda:1]

Description
Device on which to run the calculation (e.g. cpu, cuda:0).

If empty, the device can be controlled using environment variables for TensorFlow or PyTorch.

`NumThreads`

Type
String

Default value

GUI name
Number of threads

Description
Number of threads.

If not empty, OMP_NUM_THREADS will be set to this number; for PyTorch-engines, torch.set_num_threads() will be called.

Note: Because the calculation runs in a separate process, the number of threads is controlled by the input keyword NumThreads and *not* by the environment variable NSCM. **We recommend setting NSCM=1** when using the MLPotential engine.

Only single-node calculations are currently supported.

SLURM users may need to set SCM_DISABLE_MPI=1.

AMS will report the compute resources it found for machine learning potentials in standard output after applying any restrictions from environment variables as well as `NumThreads` and `Device`.

GPU with PyTorch backend:

```
#####################PyTorch setup###################
Number of threads were not limited, using all available CPU cores.
Using GPU: "cuda:0" as found by auto setup.

#####################################################
```

GPU with TensorFlow backend:

```
###################TensorFlow setup#################
TensorFlow found the following devices:
PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU')
PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')
NumThreads was not specified in the MLPotential engine block, so TensorFlow will use␣
↪all available cores.
#####################################################
```

CPU only with PyTorch backend:

```
#####################PyTorch setup###################
Number of threads were not limited, using all available CPU cores.
Using CPU only.
#####################################################
```

CPU only with TensorFlow backend:

```
###################TensorFlow setup#################
TensorFlow found the following devices:
PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU')
NumThreads was not specified in the MLPotential engine block, so TensorFlow will use␣
↪all available cores.
#####################################################
```

CHAPTER

FIVE

EXAMPLES

Before running these examples, you need to *install* (page 5) the appropriate backends.

See also the Python example for M3GNet.

5.1 AIMNet2

```
#!/bin/sh

export NSCM=1

"$AMSBIN/ams" --delete-old-results << eor

Task SinglePoint
Properties
    Gradients Yes
    DipoleMoment Yes
    Charges Yes
End
System
    Atoms
        C -3.1168071304396 -3.303815558265976 0.0
        C -1.981697818585937 -4.344278296437102 -0.02375164458614451
        H -2.786224865835057 -2.422406202569603 0.5293702995256127
        H -3.978089852534168 -3.72201295092763 0.4997029251388921
        H -3.382157359184612 -3.037352354271593 -1.012416227162533
        H -1.120415096491369 -3.926080903775447 -0.5234545697250367
        H -2.312280083190481 -5.225687652133474 -0.5531219441117571
        H -1.716347589840925 -4.610741500431485 0.9886645825763885
    End
    Charge 0
End

Engine MLPotential
    Model AIMNet2-B973c
EndEngine
eor
```

5.2 M3GNet-UP-2022

```
#!/bin/sh

export NSCM=1

"$AMSBIN/ams" <<EOF
Task SinglePoint

Properties
    Gradients True
    StressTensor Yes
End

System
    Atoms
        C 1.1705 0.2287 -0.3792
        C 0.0000 -0.6414 0.0001
        C -1.1706 0.2288 0.3792
        O -2.2331 0.0875 -0.1770
        O 2.2330 0.0876 0.1771
        H 1.0604 0.9753 -1.1517
        H -0.2743 -1.2708 -0.8464
        H 0.2743 -1.2705 0.8469
        H -1.0592 0.9787 1.1484
    End
    Lattice
       7.0000 0.0000 0.0000
       0.0000 7.0000 0.0000
       0.0000 0.0000 7.0000
    End
End

Engine MLPotential
    Model M3GNET-UP-2022
EndEngine

EOF
```

5.3 MACE

```
#!/bin/sh

export NSCM=1

$AMSBIN/ams --delete-old-results << eor
Task GeometryOptimization

GeometryOptimization
    OptimizeLattice Yes
End

System
    Atoms
```

(continues on next page)

(continued from previous page)

```
        Ti 0.0 0.0 0.0
        Ti 2.295 2.295 1.48
        O 3.672 0.918 1.48
        O 0.918 3.672 1.48
        O 1.377 1.377 0.0
        O -1.377 -1.377 0.0
    End
    Lattice
        4.59 0.0 0.0
        0.0 4.59 0.0
        0.0 0.0 2.96
    End
End

Engine MLPotential
    Backend MACE
    Model MACE-MPA-0
EndEngine
eor
```

```
#!/bin/sh

export NSCM=1

$AMSBIN/ams --delete-old-results <<EOF
Task SinglePoint

System
    Atoms
        Fe 1.668114040730352 0.1262520819109224 -0.2724347551151446
        C 2.874773495523232 0.1245428246154846 1.415559348633178
        C 2.864258443334903 0.1209340150334762 -1.968408033283078
        C 2.045488061455903 -1.017035067242245 1.417327562237913
        C 2.045319712526689 1.266012877626015 1.420696387814215
        C 2.034996210818667 1.262513524978156 -1.970493441468708
        C 2.03481947582235 -1.020501448939123 -1.962197066617627
        C 0.7035387107772729 -0.5811263243586964 1.423538567180201
        C 0.7034224460425285 0.8298589950221958 1.425623912376544
        C 0.6930135800564622 0.8266219565708358 -1.965565890130913
        C 0.6929106632695451 -0.5843873011438833 -1.960428837552032
        H 3.953018521369533 0.1247117414887672 1.357511955640776
        H 3.942916839876545 0.1210957019063513 -1.918153717773214
        H 2.378591921119416 -2.042633565448724 1.360726469795222
        H 2.378211455132701 2.291885187385673 1.368152852262757
        H 2.368487649252263 2.288431948587419 -1.922729549563146
        H 2.368164382464549 -2.046021823720774 -1.905595942151872
        H -0.1692093008491947 -1.215015035830631 1.373284230500532
        H -0.1693975768578689 1.463823066120819 1.377860052750967
        H -0.1792940809531387 1.460901281003479 -1.913022353631542
        H -0.1795070909127082 -1.218027215565519 -1.902381441905027
    End
End

Engine MLPotential
    Backend MACE
    Model Custom
```

(continues on next page)

(continued from previous page)

```
    ParameterFile $AMSHOME/examples/MLPotential/MACE-Custom-MH-1/mace-mh-1.model
    MACE
        DataType float64
        ModelHead OMOL
    End
EndEngine

EOF
```

Note: The `MACE-MH-1` model is licensed under ASL (https://github.com/gabor1/ASL/blob/main/ASL.md). This means it can only be used free-of-charge in an academic (non-commercial) setting.

5.4 NequIP

```
#!/bin/sh

export NSCM=1

$AMSBIN/ams --delete-old-results <<EOF
Task SinglePoint

Properties
    Gradients Yes
End

System
    Atoms
        O -1.239192986364078 0.7447630849670073 0.2053776714668624
        C -0.4264399234121155 -0.3765076809707508 -0.005559613642175442
        C 1.015520464999812 -0.00839542074865508 0.1578564463228227
        O 1.445056954461275 0.3261487267553756 1.250054826420532
        H -1.187895927300546 1.292145824840463 -0.6216000200358711
        H -0.6908065725149979 -1.158618889736239 0.7362545961529634
        H -0.5963066387066123 -0.7970408421468641 -1.022241368854639
        H 1.680064628837262 -0.02249480296033578 -0.7001425378304952
    End
End

Engine MLPotential
    Backend NequIP
    Model Custom
    ParameterFile $AMSHOME/examples/MLPotential/NequIP-Custom/model.pth
    MLEnergyUnit eV
    MLDistanceUnit angstrom
EndEngine
EOF
```

5.5 UMA

```
#!/bin/sh

export NSCM=1

"$AMSBIN/ams" --delete-old-results << eor
Task GeometryOptimization

System
  Atoms
    O 0. 0. -0.6
    H 0. 0. 0.6
  End
  Charge -1
End

Engine MLPotential
    Model UMA-S-1.2-OMol
EndEngine
eor
```

```
#!/bin/sh

export NSCM=1

for unpaired_e in 0 2; do

AMS_JOBNAME=o2_unpaired_e_${unpaired_e} $AMSBIN/ams --delete-old-results -n 1 <<eor
Task GeometryOptimization
System
  Atoms
    O 0. 0. -0.6
    O 0. 0. 0.6
  End
End

Engine MLPotential
    Model UMA-S-1.2-OMol
    Unrestricted Yes
    UnpairedElectrons ${unpaired_e}
EndEngine
eor

done
```

CHAPTER

SIX

AMS DRIVER'S TASKS AND PROPERTIES

MLPotential is an engine used by the AMS driver. While the specific options for the MLPotential engine are described in this manual, the definition of the system, the selection of the task and certain (potential-energy-surface-related) properties are documented in the AMS driver's manual.

On this page, you will find useful links to the relevant sections of the AMS driver's Manual.

6.1 Geometry, System definition

The definition of the system, i.e. the atom types and atomic coordinates (and optionally, the lattice vectors and atomic masses for isotopes) are part of the AMS driver input. See the System definition section of the AMS manual.

Note: The MLPotential engine currently only supports 0D (molecules) and 3D (bulk) systems.

6.2 Tasks: exploring the PES

The job of the AMS driver is to handle all changes in the simulated system's geometry, e.g. during a geometry optimization or molecular dynamics calculation, using energy and forces calculated by the engine.

These are the tasks available in the AMS driver:

- GCMC (Grand Canonical Monte Carlo)
- Geometry Optimization
- IRC (Intrinsic Reaction Coordinate)
- Molecular Dynamics
- NEB (Nudged Elastic Band)
- PESScan (Potential Energy Surface Scan, including linear transit)
- Single Point
- Transition State Search
- Vibrational Analysis

6.3 Properties in the AMS driver

The following properties can be requested from the MLPotential engine in the AMS driver's input:

- Elastic tensor
- Hessian
- Nuclear gradients (forces)
- Normal modes
- PES point character
- Phonons
- Stress tensor
- Thermodynamic properties

CHAPTER

SEVEN

FREQUENTLY ASKED QUESTIONS

7.1 General

7.1.1 Can I train my own ML potentials?

Yes, see documentation pages for ParAMS and Simple Active Learning.

7.1.2 Can I use implicit solvation (e.g., COSMO) with ML potentials?

No.

7.1.3 Do you use FAIRChem v1 or v2 for the UMA models?

It is FAIRChem v2. The AMS package manager lists the version number as "1.0", but that includes FAIRChem v2.

7.2 Errors and warning messages

7.2.1 Isolated atoms found in the structure

This message can appear with M3GNet when there are isolated atoms (atoms with no neighbors) in the structure. If this is what you expect, then you can ignore the message.

The warning may also appear as "stderr: Isolated atoms found in the structure".

7.2.2 sh: line 1: 1351 Illegal instruction: 4 sh

You may be attempting to run PyTorch on a rather old CPU. Make sure you have installed PyTorch and all other packages through the AMS package manager.

CHAPTER

EIGHT

MLPOTENTIAL KEYWORDS

8.1 Engine MLPotential

Backend

Type
Multiple Choice

Options
[FAIRChem, M3GNet, MACE, NequIP, TorchANI]

Description
The machine learning potential backend.

Device

Type
Multiple Choice

Default value
Options
[, cpu, cuda:0, cuda:1]

Description
Device on which to run the calculation (e.g. cpu, cuda:0).

If empty, the device can be controlled using environment variables for TensorFlow or PyTorch.

FAIRChem

Type
Block

Recurring
False

Description
Options for the FAIRChem machine learning potential backend.

ModelTask

Type
String

Default value
Description
Model task to use if a custom UMA/eSEN model is supplied via a parameter file (e.g. 'OC20',

'OC22', 'OC25', 'ODAC', 'OMat', 'OMC', 'OMol'). Ignored if a specific FAIRChem model is selected.

MACE

Type
Block

Recurring
False

Description
Options for the MACE machine learning potential backend.

DataType

Type
Multiple Choice

Default value
float32

Options
[float32, float64]

Description
Using `float32` is faster but less accurate, and generally recommended for MD. Conversely using `float64` is slower but more accurate, and recommended for geometry optimization.

EnableCuEquivariance

Type
Bool

Default value
Yes

Description
Enable CUDA-accelerated cuEquivariance library for equivariant neural networks, if CUDA available.

ModelHead

Type
String

Default value
Description
Model head to use if a custom MACE model is supplied via a parameter file (e.g. 'omat_pbe', 'omol', 'spice_wB97M', 'ωB97M-D3(BJ)', 'rgd1_b3lyp', 'oc20_usemppbe', 'matpes_r2scan'). Ignored if a specific MACE model is selected.

MLDistanceUnit

Type
Multiple Choice

Default value
Auto

Options
[Auto, angstrom, bohr]

GUI name
Internal distance unit

Description
Unit of distances expected by the ML backend (not the ASE calculator). The ASE calculator may require this information.

`MLEnergyUnit`

Type
Multiple Choice

Default value
Auto

Options
[Auto, Hartree, eV, kcal/mol, kJ/mol]

GUI name
Internal energy unit

Description
Unit of energy output by the ML backend (not the unit output by the ASE calculator). The ASE calculator may require this information.

`Model`

Type
Multiple Choice

Default value
ANI-2x

Options
[Custom, AIMNet2-B973c, AIMNet2-wB97MD3, ANI-1ccx, ANI-1x, ANI-2x, eSEN-S-Con-OMol, M3GNet-UP-2022, MACE-MP-0-Large, MACE-MP-0-Medium, MACE-MP-0-Small, MACE-MPA-0, UMA-S-1.2-OC20, UMA-S-1.2-OC22, UMA-S-1.2-OC25, UMA-S-1.2-ODAC, UMA-S-1.2-OMat, UMA-S-1.2-OMC, UMA-S-1.2-OMol]

Description
Select a pre-parameterized or custom model.

AIMNet2-(wB97MD3/B973c): best for fast calculations of small, drug-like molecules; limited to aperiodic systems of 14 elements (H, B, C, N, O, F, Si, P, S, Cl, As, Se, Br, I).

ANI-(1x/1ccx/2x): best for very fast calculations of organic molecules; limited to elements H, C, N, O (ANI-1x/1ccx), F, S, Cl (ANI-2x).

eSEN-S-Con-OMol: best for highly accurate calculations of diverse organic and bio-relevant molecules; not intended for calculations on periodic inorganic bulk materials.

M3GNet-UP-2022: best for fast calculations of inorganic crystalline materials; not designed for accurately modeling small organic molecules or biomolecules.

MACE-MP-0-(Small/Medium/Large): best for accurate periodic calculations of inorganic materials; size trades speed/accuracy; not designed for accurately modeling small organic molecules or biomolecules. MACE-MPA-0 has improved accuracy vs MP-0.

UMA-S-1.2 variants: best for high accuracy calculations on a broad range of systems; choose from OC20 (adsorption and surface chemistry), OC22 (oxide catalysis), OC25 (electrocatalysis), ODAC (adsorption in porous frameworks), OMat (inorganic materials), OMC (organic molecular crystals), OMol (molecules, biomolecules, metal complexes, electrolytes); can be computationally expensive compared to other, more targeted models.

Set Custom to choose a backend and provide your own parameters.

NumThreads

Type
String

Default value
GUI name
Number of threads

Description
Number of threads.

If not empty, OMP_NUM_THREADS will be set to this number; for PyTorch-engines, torch.set_num_threads() will be called.

ParameterDir

Type
String

Default value
GUI name
Parameter directory

Description
Path to a set of parameters for the backend, if it expects to read from a directory.

ParameterFile

Type
String

Default value
Description
Path to a set of parameters for the backend, if it expects to read from a file.

UnpairedElectrons

Type
Integer

Default value
0

Value Range
value >= 0

GUI name
Spin polarization

Description
The number of unpaired electrons in the system for a spin unrestricted calculation. The spin multiplicity is taken as this value plus one.

Unrestricted

Type
Bool

Default value
No

Description

Enables spin unrestricted calculations, passing spin information to the machine learning model. Only applicable to 'UMA-S-1.2-OMol', 'eSEN-S-Con-OMol' and custom FAIRChem models.

CHAPTER

NINE

KF OUTPUT FILES

9.1 Accessing KF files

KF files are Direct Access binary files. KF stands for Keyed File: KF files are keyword oriented, which makes them easy to process by simple procedures. Internally all the data on KF files is organized into sections containing variables, so each datum on the file can be identified by the combination of section and variable.

All KF files can be opened using the KFbrowser GUI program:

```
$AMSBIN/kfbrowser path/to/ams.rkf
```

By default KFbrowser shows a just a curated summary of the results on the file, but you can make it show the raw section and variable structure by switching it to expert mode. To do this, click on **File → Expert Mode** or press **ctrl/cmd + e**.

KF files can be opened and read with Command line tools.

For working with the data from KF files, it is often useful to be able to read them from Python. Using the AMS Python Stack, this can easily be done with the AKFReader class:

```
>>> from scm.akfreader import AKFReader
>>> kf = AKFReader("path/to/ams.rkf")
>>> "Molecule%Coords" in kf
True
>>> kf.description("Molecule%Coords")
{
    '_type': 'float_array',
    '_shape': [3, 'nAtoms'],
    '_comment': 'Coordinates of the nuclei (x,y,z)',
    '_unit': 'Bohr'
}
>>> kf.read("Molecule%Coords")
array([[-11.7770694 ,  -4.19739597,   0.04934546],
       [ -9.37471321,  -2.63234227,  -0.13448698],
        ...
       [ 10.09508738,  -1.06191208,   1.45286913],
       [ 10.11689333,  -1.5080196 ,  -1.87916127]])
```

Tip: For a full overview of the available methods in AKFReader, see the AKFReader API documentation.

9.2 Sections and variables on mlpotential.rkf

9.2.1 AMSResults

KF Section: AMSResults

Content: Generic results of the MLPotential evaluation.

AMSResults%AtomicDipoleMoments

Type
float_array

Description
Atomic dipole moments computed by the engine.

Unit
e*bohr

Shape
[3, Molecule%nAtoms]

AMSResults%Bonds

Type
subsection

Description
Bond info

AMSResults%Bonds.Atoms

Type
archived_int_array

Description
?

AMSResults%Bonds.CellShifts

Type
archived_int_array

Description
?

AMSResults%Bonds.description

Type
string

Description
A string containing a description of how the bond orders were calculated / where they come from

AMSResults%Bonds.hasCellShifts

Type
bool

Description
Whether there are cell shifts (relevant only in case of periodic boundary conditions)

AMSResults%Bonds.Index

Type
archived_int_array

Description
index(i) points to the first element of Atoms, Orders, and CellShifts belonging to bonds from atom 'i'. Index(1) is always 1, Index(nAtoms+1) is always nBonds + 1

AMSResults%Bonds.nLattVec

Type
int

Description
Number of lattice vectors (0:molecule, 1:chain, 2:slab, 3:bulk). This determines how the lattice displacements for bonds are interpreted.

AMSResults%Bonds.Orders

Type
archived_float_array

Description
The bond orders.

AMSResults%BulkModulus

Type
float

Description
The Bulk modulus (conversion factor from hartree/bohr^3 to GPa: 29421.026)

Unit
hartree/bohr^3

AMSResults%Charges

Type
float_array

Description
Net atomic charges as computed by the engine (for example, the Charges for a water molecule might be [-0.6, 0.3, 0.3]). The method used to compute these atomic charges depends on the engine.

Unit
e

Shape
[Molecule%nAtoms]

AMSResults%DipoleGradients

Type
float_array

Description
Derivative of the dipole moment with respect to nuclear displacements.

Shape
[3, 3, Molecule%nAtoms]

AMSResults%DipoleMoment

Type
float_array

Description
Dipole moment vector (x,y,z)

Unit
e*bohr

Shape
[3]

AMSResults%ElasticTensor

Type
float_array

Description
The elastic tensor in Voigt notation (6x6 matrix for 3D periodic systems, 3x3 matrix for 2D periodic systems, 1x1 matrix for 1D periodic systems).

Unit
hartree/bohr^nLatticeVectors

Shape
[:, :]

AMSResults%Energy

Type
float

Description
The energy computed by the engine.

Unit
hartree

AMSResults%EnergyUncertainty

Type
float

Description
Uncertainty in the energy predicted by the engine. Exact meaning depends on the engine used.

Unit
Hartree

AMSResults%Gradients

Type
float_array

Description
The nuclear gradients.

Unit
hartree/bohr

Shape
[3, Molecule%nAtoms]

AMSResults%GradientsMagnitudeUncertainty

Type
float_array

Description
Uncertainty in the magnitude of the gradients based on the variance formula (error propagation).

Unit
Hartree/Bohr

Shape
[Molecule%nAtoms]

AMSResults%GradientsUncertainty

Type
float_array

Description
Uncertainty in the nuclear gradients predicted by the engine. Exact meaning depends on the engine used.

Unit
Hartree/Bohr

Shape
[3, Molecule%nAtoms]

AMSResults%Hessian

Type
float_array

Description
The Hessian matrix

Unit
hartree/bohr^2

Shape
[3*Molecule%nAtoms, 3*Molecule%nAtoms]

AMSResults%Molecules

Type
subsection

Description
Molecules

AMSResults%Molecules.AtCount

Type
archived_int_array

Description
shape=(nMolType), Summary: number of atoms per formula.

AMSResults%Molecules.Atoms

Type
archived_int_array

Description
shape=(nAtoms), atoms(index(i):index(i+1)-1) = atom indices of molecule i

AMSResults%Molecules.Count

Type
archived_int_array

Description
Mol count per formula.

AMSResults%Molecules.Formulas

Type
string

Description
Summary: unique molecule formulas

AMSResults%Molecules.Index

Type
archived_int_array

Description
shape=(nMol+1), index(i) = index of the first atom of molecule i in array atoms(:)

AMSResults%Molecules.Type

Type
archived_int_array

Description
shape=(nMol), type of the molecule, reference to the summary arrays below

AMSResults%PESPointCharacter

Type
string

Description
The character of a PES point.

Possible values
['local minimum', 'transition state', 'stationary point with >1 negative frequencies', 'non-stationary point']

AMSResults%PoissonRatio

Type
float

Description
The Poisson ratio

AMSResults%ShearModulus

Type
float

Description
The Shear modulus (conversion factor from hartree/bohr^3 to GPa: 29421.026)

Unit
hartree/bohr^3

AMSResults%StressTensor

Type
float_array

Description
The clamped-ion stress tensor in Cartesian notation.

Unit
hartree/bohr^nLatticeVectors

Shape
[:, :]

`AMSResults%YoungModulus`

Type
float

Description
The Young modulus (conversion factor from hartree/bohr^3 to GPa: 29421.026)

Unit
hartree/bohr^3

9.2.2 BZcell(primitive cell)

KF Section: BZcell(primitive cell)
Content: The Brillouin zone of the primitive cell.

`BZcell(primitive cell)%boundaries`

Type
float_array

Description
Normal vectors for the boundaries.

Shape
[ndim, nboundaries]

`BZcell(primitive cell)%distances`

Type
float_array

Description
Distance to the boundaries.

Shape
[nboundaries]

`BZcell(primitive cell)%idVerticesPerBound`

Type
int_array

Description
The indices of the vertices per bound.

Shape
[nvertices, nboundaries]

`BZcell(primitive cell)%latticeVectors`

Type
float_array

Description
The lattice vectors.

Shape
[3, :]

BZcell(primitive cell)%nboundaries

Type
int

Description
The nr. of boundaries for the cell.

BZcell(primitive cell)%ndim

Type
int

Description
The nr. of lattice vectors spanning the Wigner-Seitz cell.

BZcell(primitive cell)%numVerticesPerBound

Type
int_array

Description
The nr. of vertices per bound.

Shape
[nboundaries]

BZcell(primitive cell)%nvertices

Type
int

Description
The nr. of vertices of the cell.

BZcell(primitive cell)%vertices

Type
float_array

Description
The vertices of the bounds.

Unit
a.u.

Shape
[ndim, nvertices]

9.2.3 DOS_Phonons

KF Section: DOS_Phonons

Content: Phonon Density of States

`DOS_Phonons%DeltaE`

Type
float

Description
The energy difference between sampled DOS energies. When there is no DOS at all a certain energy range can be skipped.

Unit
hartree

`DOS_Phonons%Energies`

Type
float_array

Description
The energies at which the DOS is sampled.

Unit
hartree

Shape
[nEnergies]

`DOS_Phonons%Fermi Energy`

Type
float

Description
The fermi energy.

Unit
hartree

`DOS_Phonons%IntegrateDeltaE`

Type
bool

Description
If enabled it means that the DOS is integrated over intervals of DeltaE. Sharp delta function like peaks cannot be missed this way.

`DOS_Phonons%nEnergies`

Type
int

Description
The nr. of energies to use to sample the DOS.

`DOS_Phonons%nSpin`

Type
int

Description
The number of spin components for the DOS.

Possible values
[1, 2]

`DOS_Phonons%Total DOS`

Type
float_array

Description
The total DOS.

Shape
[nEnergies, nSpin]

9.2.4 General

KF Section: General
Content: General information about the MLPotential calculation.

`General%account`

Type
string

Description
Name of the account from the license

`General%engine input`

Type
string

Description
The text input of the engine.

`General%engine messages`

Type
string

Description
Message from the engine. In case the engine fails to solves, this may contains extra information on why.

`General%file-ident`

Type
string

Description
The file type identifier, e.g. RKF, RUNKF, TAPE21…

`General%jobid`

Type
int

Description
Unique identifier for the job.

General%program

Type
string

Description
The name of the program/engine that generated this kf file.

General%release

Type
string

Description
The version of the program that generated this kf file (including svn revision number and date).

General%termination status

Type
string

Description
The termination status. Possible values: 'NORMAL TERMINATION', 'NORMAL TERMINATION with warnings', 'NORMAL TERMINATION with errors', 'ERROR', 'IN PROGRESS'.

General%title

Type
string

Description
Title of the calculation.

General%uid

Type
string

Description
SCM User ID

General%version

Type
int

Description
Version number?

9.2.5 KFDefinitions

KF Section: KFDefinitions
Content: The definitions of the data on this file

KFDefinitions%json

Type
string

Description
The definitions of the data on this file in json.

9.2.6 kspace(primitive cell)

KF Section: kspace(primitive cell)

Content: should not be here!!!

`kspace(primitive cell)%avec`

Type
float_array

Description
The lattice stored as a 3xnLatticeVectors matrix. Only the ndimk,ndimk part has meaning.

Unit
bohr

Shape
[3, :]

`kspace(primitive cell)%bvec`

Type
float_array

Description
The inverse lattice stored as a 3x3 matrix. Only the ndimk,ndimk part has meaning.

Unit
1/bohr

Shape
[ndim, ndim]

`kspace(primitive cell)%kt`

Type
int

Description
The total number of k-points used by the k-space to sample the unique wedge of the Brillouin zone.

`kspace(primitive cell)%kuniqu`

Type
int

Description
The number of symmetry unique k-points where an explicit diagonalization is needed. Smaller or equal to kt.

`kspace(primitive cell)%ndim`

Type
int

Description
The nr. of lattice vectors.

`kspace(primitive cell)%ndimk`

Type
int

Description
The nr. of dimensions used in the k-space integration.

`kspace(primitive cell)%xyzpt`

Type
float_array

Description
The coordinates of the k-points.

Unit
1/bohr

Shape
[ndimk, kt]

9.2.7 Low Frequency Correction

KF Section: Low Frequency Correction
Content: Configuration for the Head-Gordon Dampener-powered Free Rotor Interpolation.

`Low Frequency Correction%Alpha`

Type
float

Description
Exponent term for the Head-Gordon dampener.

`Low Frequency Correction%Frequency`

Type
float

Description
Frequency around which interpolation happens, in 1/cm.

`Low Frequency Correction%Moment of Inertia`

Type
float

Description
Used to make sure frequencies of less than ca. 1 1/cm don't overestimate entropy, in kg m^2.

9.2.8 Mobile Block Hessian

KF Section: Mobile Block Hessian
Content: Mobile Block Hessian.

`Mobile Block Hessian%Coordinates Internal`

Type
float_array

Description
?

`Mobile Block Hessian%Free Atom Indexes Input`

Type
int_array

Description
?

Mobile Block Hessian%Frequencies in atomic units

Type
float_array

Description
?

Mobile Block Hessian%Frequencies in wavenumbers

Type
float_array

Description
?

Mobile Block Hessian%Input Cartesian Normal Modes

Type
float_array

Description
?

Mobile Block Hessian%Input Indexes of Block #

Type
int_array

Description
?

Mobile Block Hessian%Intensities in km/mol

Type
float_array

Description
?

Mobile Block Hessian%MBH Curvatures

Type
float_array

Description
?

Mobile Block Hessian%Number of Blocks

Type
int

Description
Number of blocks.

Mobile Block Hessian%Sizes of Blocks

Type
int_array

Description
Sizes of the blocks.

Shape
[Number of Blocks]

9.2.9 Molecule

KF Section: Molecule
Content: The input molecule of the calculation.

`Molecule%AtomicNumbers`

Type
int_array

Description
Atomic number 'Z' of the atoms in the system

Shape
[nAtoms]

`Molecule%AtomMasses`

Type
float_array

Description
Masses of the atoms

Unit
a.u.

Values range
[0, '\infinity']

Shape
[nAtoms]

`Molecule%AtomSymbols`

Type
string

Description
The atom's symbols (e.g. 'C' for carbon)

Shape
[nAtoms]

`Molecule%bondOrders`

Type
float_array

Description
The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

`Molecule%Charge`

Type
float

Description
Net charge of the system

Unit
e

Molecule%Coords

Type
float_array

Description
Coordinates of the nuclei (x,y,z)

Unit
bohr

Shape
[3, nAtoms]

Molecule%eeAttachTo

Type
int_array

Description
UNUSED IN AMS>=2026. A multipole may be attached to an atom. This influences the energy gradient.

Molecule%eeChargeWidth

Type
float

Description
If charge broadening was used for external charges, this represents the width of the charge distribution.

Molecule%eeEField

Type
float_array

Description
The external homogeneous electric field.

Unit
hartree/(e*bohr)

Shape
[3]

Molecule%eeLatticeVectors

Type
float_array

Description
UNUSED IN AMS>=2026. The lattice vectors used for the external point- or multipole-charges.

Unit
bohr

Shape
[3, eeNLatticeVectors]

`Molecule%eeMulti`

Type
float_array

Description
The values of the external point- or multipole- charges.

Unit
a.u.

Shape
[eeNZlm, eeNMulti]

`Molecule%eeNLatticeVectors`

Type
int

Description
UNUSED IN AMS>=2026. The number of lattice vectors for the external point- or multipole-charges.

`Molecule%eeNMulti`

Type
int

Description
The number of external point- or multipole- charges.

`Molecule%eeNZlm`

Type
int

Description
When external point- or multipole- charges are used, this represents the number of spherical harmonic components. E.g. if only point charges were used, eeNZlm=1 (s-component only). If point charges and dipole moments were used, eeNZlm=4 (s, px, py and pz).

`Molecule%eeUseChargeBroadening`

Type
bool

Description
Whether or not the external charges are point-like or broadened.

`Molecule%eeXYZ`

Type
float_array

Description
The position of the external point- or multipole- charges.

Unit
bohr

Shape
[3, eeNMulti]

Molecule%EngineAtomicInfo

Type
string_fixed_length

Description
Atom-wise info possibly used by the engine.

Molecule%fromAtoms

Type
int_array

Description
Index of the first atom in a bond. See the bondOrders array

Molecule%latticeDisplacements

Type
int_array

Description
The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

Molecule%LatticeVectors

Type
float_array

Description
Lattice vectors

Unit
bohr

Shape
[3, nLatticeVectors]

Molecule%nAtoms

Type
int

Description
The number of atoms in the system

Molecule%nAtomsTypes

Type
int

Description
The number different of atoms types

Molecule%nLatticeVectors

Type
int

Description
Number of lattice vectors (i.e. number of periodic boundary conditions)

Possible values
[0, 1, 2, 3]

`Molecule%toAtoms`

Type
int_array

Description
Index of the second atom in a bond. See the bondOrders array

9.2.10 MoleculeSuperCell

KF Section: MoleculeSuperCell
Content: The system used for the numerical phonon super cell calculation.

`MoleculeSuperCell%AtomicNumbers`

Type
int_array

Description
Atomic number 'Z' of the atoms in the system

Shape
[nAtoms]

`MoleculeSuperCell%AtomMasses`

Type
float_array

Description
Masses of the atoms

Unit
a.u.

Values range
[0, '\infinity']

Shape
[nAtoms]

`MoleculeSuperCell%AtomSymbols`

Type
string

Description
The atom's symbols (e.g. 'C' for carbon)

Shape
[nAtoms]

`MoleculeSuperCell%bondOrders`

Type
float_array

Description
The bond orders for the bonds in the system. The indices of the two atoms participating in

the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

MoleculeSuperCell%Charge

Type
float

Description
Net charge of the system

Unit
e

MoleculeSuperCell%Coords

Type
float_array

Description
Coordinates of the nuclei (x,y,z)

Unit
bohr

Shape
[3, nAtoms]

MoleculeSuperCell%eeAttachTo

Type
int_array

Description
UNUSED IN AMS>=2026. A multipole may be attached to an atom. This influences the energy gradient.

MoleculeSuperCell%eeChargeWidth

Type
float

Description
If charge broadening was used for external charges, this represents the width of the charge distribution.

MoleculeSuperCell%eeEField

Type
float_array

Description
The external homogeneous electric field.

Unit
hartree/(e*bohr)

Shape
[3]

MoleculeSuperCell%eeLatticeVectors

Type
float_array

Description
UNUSED IN AMS>=2026. The lattice vectors used for the external point- or multipole-charges.

Unit
bohr

Shape
[3, eeNLatticeVectors]

MoleculeSuperCell%eeMulti

Type
float_array

Description
The values of the external point- or multipole- charges.

Unit
a.u.

Shape
[eeNZlm, eeNMulti]

MoleculeSuperCell%eeNLatticeVectors

Type
int

Description
UNUSED IN AMS>=2026. The number of lattice vectors for the external point- or multipole-charges.

MoleculeSuperCell%eeNMulti

Type
int

Description
The number of external point- or multipole- charges.

MoleculeSuperCell%eeNZlm

Type
int

Description
When external point- or multipole- charges are used, this represents the number of spherical harmonic components. E.g. if only point charges were used, eeNZlm=1 (s-component only). If point charges and dipole moments were used, eeNZlm=4 (s, px, py and pz).

MoleculeSuperCell%eeUseChargeBroadening

Type
bool

Description
Whether or not the external charges are point-like or broadened.

MoleculeSuperCell%eeXYZ

Type
float_array

Description
The position of the external point- or multipole- charges.

Unit
bohr

Shape
[3, eeNMulti]

MoleculeSuperCell%EngineAtomicInfo

Type
string_fixed_length

Description
Atom-wise info possibly used by the engine.

MoleculeSuperCell%fromAtoms

Type
int_array

Description
Index of the first atom in a bond. See the bondOrders array

MoleculeSuperCell%latticeDisplacements

Type
int_array

Description
The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

MoleculeSuperCell%LatticeVectors

Type
float_array

Description
Lattice vectors

Unit
bohr

Shape
[3, nLatticeVectors]

MoleculeSuperCell%nAtoms

Type
int

Description
The number of atoms in the system

MoleculeSuperCell%nAtomsTypes

Type
int

Description
The number different of atoms types

MoleculeSuperCell%nLatticeVectors

Type
int

Description
Number of lattice vectors (i.e. number of periodic boundary conditions)

Possible values
[0, 1, 2, 3]

`MoleculeSuperCell%toAtoms`

Type
int_array

Description
Index of the second atom in a bond. See the bondOrders array

9.2.11 Other

KF Section: Other
Content: Contains any information send over by ASE/python which AMS does not know how to handle. This is stored but not documented.

9.2.12 phonon_curves

KF Section: phonon_curves
Content: Phonon dispersion curves.

`phonon_curves%brav_type`

Type
string

Description
Type of the lattice.

`phonon_curves%Edge_#_bands`

Type
float_array

Description
The band energies

Shape
[nBands, nSpin, :]

`phonon_curves%Edge_#_direction`

Type
float_array

Description
Direction vector.

Shape
[nDimK]

`phonon_curves%Edge_#_kPoints`

Type
float_array

Description
Coordinates for points along the edge.

Shape
[nDimK, :]

phonon_curves%Edge_#_labels

Type
lchar_string_array

Description
Labels for begin and end point of the edge.

Shape
[2]

phonon_curves%Edge_#_lGamma

Type
bool

Description
Is gamma point?

phonon_curves%Edge_#_nKPoints

Type
int

Description
The nr. of k points along the edge.

phonon_curves%Edge_#_vertices

Type
float_array

Description
Begin and end point of the edge.

Shape
[nDimK, 2]

phonon_curves%Edge_#_xFor1DPlotting

Type
float_array

Description
x Coordinate for points along the edge.

Shape
[:]

phonon_curves%indexLowestBand

Type
int

Description
?

phonon_curves%nBands

Type
int

Description
Number of bands.

phonon_curves%nBas

Type
int

Description
Number of basis functions.

phonon_curves%nDimK

Type
int

Description
Dimension of the reciprocal space.

phonon_curves%nEdges

Type
int

Description
The number of edges. An edge is a line-segment through k-space. It has a begin and end point and possibly points in between.

phonon_curves%nEdgesInPath

Type
int

Description
A path is built up from a number of edges.

phonon_curves%nSpin

Type
int

Description
Number of spin components.

Possible values
[1, 2]

phonon_curves%path

Type
int_array

Description
If the (edge) index is negative it means that the vertices of the edge abs(index) are swapped e.g. path = (1,2,3,0,-3,-2,-1) goes though edges 1,2,3, then there's a jump, and then it goes back.

Shape
[nEdgesInPath]

phonon_curves%path_source

Type
string

Description
Source or program used to generate the path.

Possible values
['input', 'kpath', 'seekpath']

phonon_curves%path_type

Type
string

Description
?

9.2.13 Phonons

KF Section: Phonons

Content: Information on the numerical phonons (super cell) setup. NB: the reciprocal cell of the super cell is smaller than the reciprocal primitive cell.

Phonons%Modes

Type
float_array

Description
The normal modes with the translational symmetry of the super cell.

Shape
[3, nAtoms, 3, NumAtomsPrim, nK]

Phonons%nAtoms

Type
int

Description
Number of atoms in the super cell.

Phonons%nK

Type
int

Description
Number of gamma-points (of the super cell) that fit into the primitive reciprocal cell.

Phonons%NumAtomsPrim

Type
int

Description
Number of atoms in the primitive cell.

Phonons%xyzKSuper

Type
float_array

Description
The coordinates of the gamma points that fit into the primitive reciprocal cell.

Shape
[3, nK]

9.2.14 Thermodynamics

KF Section: Thermodynamics
Content: Thermodynamic properties computed from normal modes.

`Thermodynamics%Enthalpy`

Type
float_array

Description
Enthalpy.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Entropy rotational`

Type
float_array

Description
Rotational contribution to the entropy.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Entropy total`

Type
float_array

Description
Total entropy.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Entropy translational`

Type
float_array

Description
Translational contribution to the entropy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Entropy vibrational

Type
float_array

Description
Vibrational contribution to the entropy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Gibbs free Energy

Type
float_array

Description
Gibbs free energy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Heat Capacity rotational

Type
float_array

Description
Rotational contribution to the heat capacity.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Heat Capacity total

Type
float_array

Description
Total heat capacity.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Heat Capacity translational

Type
float_array

Description
Translational contribution to the heat capacity.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Heat Capacity vibrational

Type
float_array

Description
Vibrational contribution to the heat capacity.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Inertia direction vectors

Type
float_array

Description
Inertia direction vectors.

Shape
[3, 3]

Thermodynamics%Internal Energy rotational

Type
float_array

Description
Rotational contribution to the internal energy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Internal Energy total

Type
float_array

Description
Total internal energy.

Unit
a.u.

Thermodynamics%Internal Energy translational

Type
float_array

Description
Translational contribution to the internal energy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Internal Energy vibrational

Type
float_array

Description
Vibrational contribution to the internal energy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%lowFreqEntropy

Type
float_array

Description
Entropy contributions from low frequencies (see 'lowFrequencies').

Unit
a.u.

Shape
[nLowFrequencies]

Thermodynamics%lowFreqHeatCapacity

Type
float_array

Description
Heat capacity contributions from low frequencies (see 'lowFrequencies').

Unit
a.u.

Shape
[nLowFrequencies]

Thermodynamics%lowFreqInternalEnergy

Type
float_array

Description
Internal energy contributions from low frequencies (see 'lowFrequencies').

Unit
a.u.

Shape
[nLowFrequencies]

Thermodynamics%lowFrequencies

Type
float_array

Description
Frequencies below 20 cm^-1 (contributions from frequencies below 20 cm^-1 are not included in vibrational sums, and are saved separately to 'lowFreqEntropy', 'lowFreqInternalEnergy' and 'lowFreqInternalEnergy'). Note: this does not apply to RRHO-corrected quantities.

Unit
cm^-1

Shape
[nLowFrequencies]

Thermodynamics%Moments of inertia

Type
float_array

Description
Moments of inertia.

Unit
a.u.

Shape
[3]

Thermodynamics%nLowFrequencies

Type
int

Description
Number of elements in the array lowFrequencies.

Thermodynamics%nTemperatures

Type
int

Description
Number of temperatures.

Thermodynamics%Pressure

Type
float

Description
Pressure used.

Unit
atm

Thermodynamics%RRHOCorrectedHeatCapacity

Type
float_array

Description
Heat capacity T*S corrected using the 'low vibrational frequency free rotor interpolation corrections'.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%RRHOCorrectedInternalEnergy

Type
float_array

Description
Internal energy T*S corrected using the 'low vibrational frequency free rotor interpolation corrections'.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%RRHOCorrectedTS

Type
float_array

Description
T*S corrected using the 'low vibrational frequency free rotor interpolation corrections'.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Temperature

Type
float_array

Description
List of temperatures at which properties are calculated.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%TS

Type
float_array

Description
T*S, i.e. temperature times entropy.

Unit
a.u.

Shape
[nTemperatures]

9.2.15 Vibrations

KF Section: Vibrations

Content: Information related to molecular vibrations.

`Vibrations%ExcitedStateLifetime`

Type
float

Description
Raman excited state lifetime.

Unit
hartree

`Vibrations%ForceConstants`

Type
float_array

Description
The force constants of the vibrations.

Unit
hartree/bohr^2

Shape
[nNormalModes]

`Vibrations%Frequencies[cm-1]`

Type
float_array

Description
The vibrational frequencies of the normal modes.

Unit
cm^-1

Shape
[nNormalModes]

`Vibrations%Intensities[km/mol]`

Type
float_array

Description
The intensity of the normal modes.

Unit
km/mol

Shape
[nNormalModes]

`Vibrations%IrReps`

Type
lchar_string_array

Description
Symmetry symbol of the normal mode.

Shape
[nNormalModes]

Vibrations%ModesNorm2

Type
float_array

Description
Norms of the rigid motions.

Shape
[nNormalModes+nRigidModes]

Vibrations%ModesNorm2*

Type
float_array

Description
Norms of the rigid motions (for a given irrep…?).

Shape
[nNormalModes+nRigidModes]

Vibrations%nNormalModes

Type
int

Description
Number of normal modes.

Vibrations%NoWeightNormalMode(#)

Type
float_array

Description
?.

Shape
[3, Molecule%nAtoms]

Vibrations%NoWeightRigidMode(#)

Type
float_array

Description
?

Shape
[3, Molecule%nAtoms]

Vibrations%nRigidModes

Type
int

Description
Number of rigid modes.

Vibrations%nSemiRigidModes

Type
int

Description
Number of semi-rigid modes.

`Vibrations%PVDOS`

Type
float_array

Description
Partial vibrational density of states.

Values range
[0.0, 1.0]

Shape
[nNormalModes, Molecule%nAtoms]

`Vibrations%RamanDepolRatioLin`

Type
float_array

Description
Raman depol ratio (lin).

Shape
[nNormalModes]

`Vibrations%RamanDepolRatioNat`

Type
float_array

Description
Raman depol ratio (nat).

Shape
[nNormalModes]

`Vibrations%RamanIncidentFreq`

Type
float

Description
Raman incident light frequency.

Unit
hartree

`Vibrations%RamanIntens[A^4/amu]`

Type
float_array

Description
Raman intensities

Unit
A^4/amu

Shape
[nNormalModes]

Vibrations%ReducedMasses

Type
float_array

Description
The reduced masses of the normal modes.

Unit
a.u.

Values range
[0, '\infinity']

Shape
[nNormalModes]

Vibrations%RotationalStrength

Type
float_array

Description
The rotational strength of the normal modes.

Shape
[nNormalModes]

Vibrations%TransformationMatrix

Type
float_array

Description
?

Shape
[3, Molecule%nAtoms, nNormalModes]

Vibrations%VROACIDBackward

Type
float_array

Description
VROA Circular Intensity Differential: Backward scattering.

Unit
10^{-3}

Shape
[nNormalModes]

Vibrations%VROACIDDePolarized

Type
float_array

Description
VROA Circular Intensity Differential: Depolarized scattering.

Unit
10^-3

Shape
[nNormalModes]

Vibrations%VROACIDForward

Type
float_array

Description
VROA Circular Intensity Differential: Forward scattering.

Unit
10^-3

Shape
[nNormalModes]

Vibrations%VROACIDPolarized

Type
float_array

Description
VROA Circular Intensity Differential: Polarized scattering.

Unit
10^-3

Shape
[nNormalModes]

Vibrations%VROADeltaBackward

Type
float_array

Description
VROA Intensity: Backward scattering.

Unit
10^-3 A^4/amu

Shape
[nNormalModes]

Vibrations%VROADeltaDePolarized

Type
float_array

Description
VROA Intensity: Depolarized scattering.

Unit
10^-3 A^4/amu

Shape
[nNormalModes]

Vibrations%VROADeltaForward

Type
float_array

Description
VROA Intensity: Forward scattering.

Unit
10¯3 A^4/amu

Shape
[nNormalModes]

`Vibrations%VROADeltaPolarized`

Type
float_array

Description
VROA Intensity: Polarized scattering.

Unit
10¯3 A^4/amu

Shape
[nNormalModes]

`Vibrations%ZeroPointEnergy`

Type
float

Description
Vibrational zero-point energy.

Unit
hartree

INDEX